





Armazenando Dados Client-side

Estamos de volta!

Na última unidade, vimos como trabalhar com persistência de dados de forma local em nossas aplicações e jogos web. Vimos como usar a QueryString e o objeto SessionStorage para gravar dados localmente e controlar sessões de nossos jogos. Nesta unidade vamos nos aprofundar um pouco mais no assunto, vendo como salvar informações no computador do usuário, não somente na sessão de jogo atual, mas de forma permanente, permitindo ser reutilizada se o usuário voltar ao nosso jogo.

Conceito de Persistência de Dados

Quando estamos falando em Persistência de Dados, estamos falando em armazenar informações de forma consistente e permanente. Isto é, são informações que podem ser recuperadas a qualquer momento e que não serão perdidas quando há o encerramento da sessão.

Na especificação do HTML5, tem-se a API *Web Storage* que fornece mecanismos visando usar o JavaScript para armazenar dados nos navegadores de forma eficiente. Ela fornece duas maneiras para armazenar dados em nossas aplicações e jogos (uma delas já conhecemos):

- **sessionStorage**: permite armazenar dados para uma única sessão. Como você já aprendeu, assim que o usuário fecha a janela do navegador, os dados são perdidos.

- **localStorage**: permite armazenar dados sem data de expiração, de forma que caso o navegador seja fechado ou o computador desligado, os dados permanecem armazenados no navegador.

Usar o SessionStorage será muito útil para nossos jogos. No entanto, como podemos observar, somente usando o recurso de localStorage teremos a informação salva de forma persistente.

Persistindo Dados com LocalStorage

Armazenar e recuperar valores em localStorage são feitos da mesma forma como SessionStorage, usando pares de chave e valor. Sua sintaxe é também semelhante.

A sintaxe básica para salvar e recuperar dados em localStorage é a seguinte:

```
// Salva os dados em localStorage
localStorage.setItem("chave", valor);

// Obtém os dados de localStorage
let data = localStorage.getItem("chave");
```

Note que a sintaxe e os métodos são, de fato, semelhantes aos vistos na unidade anterior. Assim, se quiséssemos armazenar permanentemente no navegador qual o time que o jogador escolheu, faríamos:

```
// Salva o time do jogador permanentemente em localStorage
localStorage.setItem("timeJogador", "Team Instinct");
```

Isso irá salvar a *string* "Team Instinct" em localStorage com a chave identificadora de timeJogador. Então, se quiséssemos acessar os valores salvos no navegador em nosso programa, usaríamos o método getItem():

```
// Obtém o time do jogador em localStorage
let time = localStorage.getItem("timeJogador");
```

Claro que a API *Web Storage* também fornece alguns métodos para remover dados, são elas:

- removeItem("chave"): remove a informação armazenada no objeto de armazenamento, juntamente com a chave passada como argumento do método.

- clear(): use este método para esvaziar todo o objeto de armazenamento no domínio da aplicação.

Por exemplo, para remover o time do jogador de localStorage faríamos:

```
// Remove a chave permanentemente de localStorage
localStorage.removeItem("timeJogador");
```

Web Storage ainda fornece alguns eventos DOM para que possamos executar blocos de código quando alguma mudança ocorre no objeto de armazenamento *storage*. Dessa forma, podemos adicionar um evento para detectar quando houverem mudanças no objeto localStorage entre abas ou janelas separadas no mesmo computador:



```
// Aplica um evento para localStorage
window.addEventListener("storage", function(event) {
  // Quando localStorage mudar escreva no console qual
  // a chave que foi alterada
  console.log(event.key + " foi alterado");
});
```

Note que este evento não será disparado na janela atual, mas nas outras janelas abertas ou em abas no mesmo navegador.

Isso será muito útil para que você possa controlar alterações entre diferentes janelas e para que possa controlar estados e mudanças significativas que possam ocorrer quando o seu jogo estiver rodando em diferentes janelas no mesmo navegador (ou se o jogador estiver tentando burlar algo).

Quando o evento for disparado, o objeto eventualmente representa o evento ocorrendo no DOM. Ele irá carregar consigo alguns atributos que nos permitem tratar e comparar as chaves e valores que foram alterados, como:

Atributo	Descrição
key	Retorna uma String que representa a chave alterada. O atributo key é null quando a alteração é causada pelo método clear() de armazenamento.
newValue	Retorna uma String com o novo valor da chave. Este valor é null quando a alteração foi invocada pelo método clear() de armazenamento ou a chave foi removida do armazenamento.
oldValue	Retorna uma String com o valor original da chave. Este valor é null quando a chave foi adicionada recentemente e, portanto, não possui nenhum valor anterior.

Lembrando que a informação fica **gravada e acessível** somente para o domínios não compartilham o que está gravado localmente no navegador.

Mais um resumo que se finda Padawan! Nesta unidade você viu mais um recurso importante para jogos web: a possibilidade de persistir informações, isto é, de mantê-las salvas no computador do usuário. Trata-se de outra possibilidade para que nossos jogos possam continuar funcionando de forma off-line ou, caso o jogador precise sair por algum motivo, ele possa voltar para continuar a jogar.

Até a próxima!

Atividade Extra

Acesse o link a seguir para realizar experimentos com os eventos de localStorage. Abra o link em duas janelas e experimente mudar o valor contido no campo "change me" em uma delas. Você verá que na segunda janela, o evento será disparado e o campo "Waiting for data via storage event..." irá registrar o valor alterado.

- **Web Storage events**, disponível em: <https://bestvpn.org/html5demos/storage-events/>.

Experimento os exemplos apresentados para consolidar seus conhecimentos vistos nessa unidade. Faça isso clicando nos botões verdes escritos "try it yourself" (tente você mesmo).

- **API HTML Web Storage, disponível** em: <https://www.w3schools.com/html/html5_webstorage.asp>.



Referência Bibliográfica

LAWSON B.; SHARP R. **Introdução ao Html 5**. Alta Books: Rio de Janeiro. 2011.

Ir para exercício